1. **Write a program that converts Pandas DataFrame and Series into NumPy. Array.**

```
import pandas as pd
import numpy as np

# Create a sample DataFrame
data = {'A': [1, 2, 3, 4],
        'B': [5, 6, 7, 8]}
df = pd.DataFrame(data)

# Convert DataFrame to NumPy array using .values attribute
numpy_array_df = df.values

# Create a sample Series
series = pd.Series([10, 20, 30, 40])

# Convert Series to NumPy array using .values attribute
numpy_array_series = series.values

# Alternatively, you can use the to_numpy() method
numpy_array_df_alt = df.to_numpy()
numpy_array_series_alt = series.to_numpy()

# Print the NumPy arrays
print("DataFrame to NumPy Array (Using .values attribute):")
print(numpy_array_df)
print("\nSeries to NumPy Array (Using .values attribute):")
print(numpy_array_series)
print("\nDataFrame to NumPy Array (Using to_numpy() method):")
print(numpy_array_df_alt)
print("\nSeries to NumPy Array (Using to_numpy() method):")
print(numpy_array_series_alt)
```

2. **Write a program that demonstrates the column selection, column addition, and column deletion.**

```
import pandas as pd

# Create a sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David'],
   'Age': [25, 30, 22, 35],
   'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
```

```
        # Column Selection
        selected_columns = df[['Name', 'Age']]
        print("Selected Columns:")
        print(selected_columns)

        # Column Addition
        df['Salary'] = [50000, 60000, 45000, 70000]
        print("\nDataFrame After Adding 'Salary' Column:")
        print(df)

        # Column Deletion
        df.drop('City', axis=1, inplace=True)  # Remove the 'City' column
        print("\nDataFrame After Deleting 'City' Column:")
print(df)
```

This program does the following:

- It creates a sample DataFrame df with columns 'Name', 'Age', and 'City'.
- Column Selection: It selects and prints the 'Name' and 'Age' columns using double square brackets [['Name', 'Age']].
- Column Addition: It adds a new 'Salary' column to the DataFrame with sample salary values.
- Column Deletion: It removes the 'City' column from the DataFrame using the drop() method with axis=1 (for column) and inplace=True to make the change permanent.

3. **Write a program that demonstrates the row selection, row addition, and row deletion.**

```
import pandas as pd

# Create a sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David'],
   'Age': [25, 30, 22, 35],
   'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)

# Row Selection
selected_row = df.loc[1]  # Selects the second row (index 1)
print("Selected Row:")
print(selected_row)
# Row Addition
new_row = pd.Series(['Eva', 28, 'Miami'], index=['Name', 'Age', 'City'])
df = df.append(new_row, ignore_index=True)  # Adds a new row to the DataFrame
print("\nDataFrame After Adding New Row:")
print(df)
# Row Deletion
df.drop(2, inplace=True)  # Removes the third row (index 2)
print("\nDataFrame After Deleting Third Row:")
print(df)
```

4. **Get n-largest and n-smallest values from a particular column in Pandas dataFrame.**

```
import pandas as pd

# Create a sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David'],
   'Age': [25, 30, 22, 35],
   'Salary': [50000, 60000, 45000, 70000]
}
df = pd.DataFrame(data)

# Get the 2 largest values from the 'Salary' column
n_largest = df['Salary'].nlargest(2)
print("2 Largest Salaries:")
print(n_largest)

# Get the 2 smallest values from the 'Age' column
n_smallest = df['Age'].nsmallest(2)
print("\n2 Smallest Ages:")
print(n_smallest)
```

In this program:

- A sample DataFrame df is created with columns 'Name', 'Age', and 'Salary'.
- To get the n-largest values from the 'Salary' column, we use the nlargest() method with n=2. The result is stored in the n_largest variable.
- To get the n-smallest values from the 'Age' column, we use the nsmallest() method with n=2. The result is stored in the n_smallest variable.
- The program then prints the n-largest and n-smallest values from their respective columns.

## Pandas Library: Visualization

1. **Write a program which use pandas inbuilt visualization to plot following graphs:**
   - **Bar plots**
   - **Histograms**
   - **Line plots**
   - **Scatter plots**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a sample DataFrame
data = {
   'Category': ['A', 'B', 'C', 'D'],
   'Values': [10, 25, 18, 32]
}
df = pd.DataFrame(data)

# a. Bar plots
df.plot(x='Category', y='Values', kind='bar', title='Bar Plot')
plt.show()

# b. Histograms
data = {
   'Values': np.random.normal(0, 1, 1000)
}
df = pd.DataFrame(data)

df['Values'].plot(kind='hist', title='Histogram', edgecolor='black', bins=20)
plt.show()

# c. Line plots
data = {
   'X': np.arange(0, 10, 0.1),
   'Y': np.sin(np.arange(0, 10, 0.1))
}
df = pd.DataFrame(data)

df.plot(x='X', y='Y', kind='line', title='Line Plot')
plt.show()

# d. Scatter plots
data = {
   'X': np.random.rand(50),
   'Y': np.random.rand(50)
}
df = pd.DataFrame(data)

df.plot(x='X', y='Y', kind='scatter', title='Scatter Plot', c='b')
plt.show()
```

**2. Write a program to demonstrate use of groupby () method.**

```
import pandas as pd

# Create a sample DataFrame
data = {
   'Category': ['A', 'B', 'A', 'B', 'A', 'C'],
   'Values': [10, 25, 18, 32, 15, 8]
}
df = pd.DataFrame(data)

# Group the DataFrame by the 'Category' column
grouped = df.groupby('Category')

# Calculate the sum of 'Values' for each group
sum_by_category = grouped['Values'].sum()
print("Sum of Values by Category:")
print(sum_by_category)

# Calculate the mean of 'Values' for each group
mean_by_category = grouped['Values'].mean()
print("\nMean of Values by Category:")
print(mean_by_category)

# You can also use multiple columns for grouping
# For example, grouping by 'Category' and 'Values'
grouped_multiple = df.groupby(['Category', 'Values'])

# Count the number of occurrences for each combination of 'Category' and 'Values'
count_by_category_values = grouped_multiple.size()
print("\nCount by Category and Values:")
print(count_by_category_values)
```

**3. Write a program to demonstrate pandas Merging, Joining and Concatenating.**

```
import pandas as pd

# Create sample DataFrames
data1 = {
   'ID': [1, 2, 3, 4],
   'Name': ['Alice', 'Bob', 'Charlie', 'David']
}

data2 = {
   'ID': [3, 4, 5, 6],
   'Age': [25, 30, 22, 35]
}
```

```
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

# Merging DataFrames
merged_df = pd.merge(df1, df2, on='ID', how='inner')
print("Merged DataFrame (Inner Join):")
print(merged_df)

# Joining DataFrames
joined_df = df1.set_index('ID').join(df2.set_index('ID'), how='outer')
print("\nJoined DataFrame (Outer Join):")
print(joined_df)

# Concatenating DataFrames
concatenated_df = pd.concat([df1, df2], axis=0, ignore_index=True)
print("\nConcatenated DataFrame:")
print(concatenated_df)
```

4. **Creating data frames from csv and excel files.**
   a. **To perform Pandas operations on the Titanic dataset**

```
import pandas as pd

# Load the Titanic dataset from a CSV file
titanic_data = pd.read_csv('titanic.csv')

# Display the first 5 rows of the dataset
print("First 5 rows of the Titanic dataset:")
print(titanic_data.head())

# Get basic statistics of the dataset
summary_stats = titanic_data.describe()
print("\nSummary Statistics:")
print(summary_stats)

# Filter passengers who survived (Survived = 1)
survived_passengers = titanic_data[titanic_data['Survived'] == 1]
print("\nPassengers who survived:")
print(survived_passengers.head())

# Filter passengers in first class (Pclass = 1)
first_class_passengers = titanic_data[titanic_data['Pclass'] == 1]
print("\nPassengers in First Class:")
print(first_class_passengers.head())

# Group passengers by class and calculate the mean age for each class
```

```
mean_age_by_class = titanic_data.groupby('Pclass')['Age'].mean()
print("\nMean Age by Class:")
print(mean_age_by_class)

# Sort the dataset by age in descending order
sorted_by_age = titanic_data.sort_values(by='Age', ascending=False)
print("\nPassengers sorted by Age (Descending):")
print(sorted_by_age.head())

# Count the number of passengers in each class
passenger_count_by_class = titanic_data['Pclass'].value_counts()
print("\nPassenger Count by Class:")
print(passenger_count_by_class)

# Calculate the correlation between 'Fare' and 'Age' columns
correlation = titanic_data['Fare'].corr(titanic_data['Age'])
print("\nCorrelation between Fare and Age:", correlation)
```

**b. To include data visualization using Pandas for the Titanic dataset**

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Titanic dataset from a CSV file
titanic_data = pd.read_csv('titanic.csv')

# Display the first 5 rows of the dataset
print("First 5 rows of the Titanic dataset:")
print(titanic_data.head())

# Get basic statistics of the dataset
summary_stats = titanic_data.describe()
print("\nSummary Statistics:")
print(summary_stats)

# Plot a histogram of passengers' ages
plt.figure(figsize=(8, 6))
sns.histplot(titanic_data['Age'], bins=20, kde=True)
plt.title("Age Distribution of Passengers")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()

# Plot a bar chart for passenger class counts
plt.figure(figsize=(8, 6))
sns.countplot(data=titanic_data, x='Pclass')
```

```
plt.title("Passenger Class Counts")
plt.xlabel("Class")
plt.ylabel("Count")
plt.show()

# Create a heatmap to visualize the correlation between numerical columns
plt.figure(figsize=(8, 6))
correlation_matrix = titanic_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```